View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Institutional repository of Tomas Bata University Library

An Optimization Scheduler in the Intranet Grid

Petr Lukasik and Martin Sysel

Department of Computer and Communication Systems
Faculty of Applied Informatics
Tomas Bata University in Zlin
nam. T. G. Masaryka 5555, 760 01 Zlin
Czech Republic
plukasik@tajmac-zps.cz,
sysel@fai.utb.cz
http://www.fai.utb.cz

Abstract. Scheduling of processes is the basic task for a grid computing. This role is responsible for the allocation of time for computational agents. The calculation agent can include a wide range of devices, based on various types of computer systems. Is it possible to efficiently build a grid infrastructure in the company environment. The grid can be used in scientific and technical computing, as well as better load distribution of the individual computing systems and services. The scheduler is a major component of grid computing. The main task is to effectively distribute the load of the system and allocate tasks to places that are not sufficiently utilized at a given moment.
The article also focuses on the relation between conflicting parameters, which relate to the quality of the planning process. Time calculation of the optimization algorithm affects the quality of the draft plan. It has a direct impact on the total period of the job processing. In the strategy of the scheduling there is a point where extensions of time have no effect on quality of the draft of the plan but getting worse the overall runtime of the job. The aim was to compare the common metaheuristic algorithms. From the measured values to propose a methodology for determining the optimum time for planning process

Keywords: Grid, JSDL, POSIX, Precedence and Optimization Scheduler

1 Introduction

Distribution of tasks and scheduling are essential elements of a grid services. A tool that allows easy definition of the role and its distribution in the environment is a prerequisite for high-quality and user-acceptable Grid Services. The user should have a freedom as well as resources to easily tracking of their own processing. An important feature is that the Grid service has the least restrictive conditions for a successful job execution. (Type or version of software, operating system and hardware features). The user of the grid should have a certain freedom. Not to be tied up of restrictive rules, except the rules relating to information security and data processing [5, 6].

The aim of this work is the description of the scheduler. The basic functionality is to create a schedule of tasks that will meet the priority, precedence and optimization requirements for the distribution and processing of batch jobs in a grid computing environment. The result is the design of optimal scheduling time in relation to the number of jobs that are released into the processing and comparison of various scheduling strategies.

The Task is defined by using the standard JSDL (Job Submission Definition Language). JSDL is an XML-based computational specification for the management and distribution of batch jobs in a Grid environment, developed by OGF-JSDL-WG [1],[2],[9]. A current version 1.0 has also the definition of the POSIX support. JSDL includes support to define a range of computing resources (the length of processing number of threads, disk space). The parameters to be defined by the user (type of system, type of processor, memory, network bandwidth) are basic inputs for the scheduler [8].

2 Long-Term Job Scheduler

The scheduler is a major component of the Intranet Grid. Logical and temporal sequence of tasks is the primary activity for him. Is designed as a set of three main modules.

The priority scheduler is responsible for optimizing the solution of tasks based on priority. The priority of the process is an optional input parameter that must be entered by the user when the task starts. Precedence scheduler is responsible for the order of the processed parts. The optimization solver searches the best possible use of computing resources. Thereby contributes to optimizing the job time makespan. Scheduler is subdivided into three independent components.

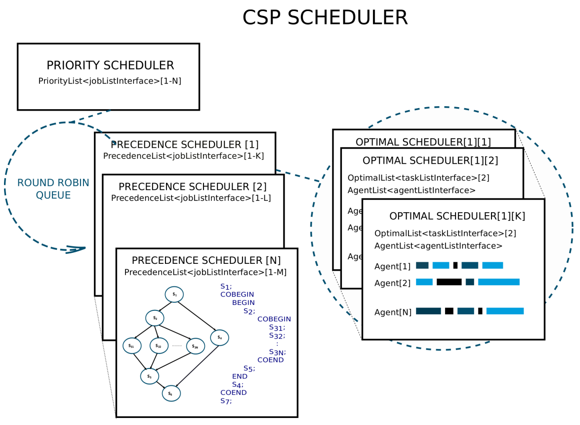
CSP SCHEDULER
PRIORITY SCHEDULER
PriorityList<jobListInterface>[1-N]
ROUND ROBIN QUEUE
PRECEDENCE SCHEDULER [1]
PrecedenceList<jobListInterface>[1-K]
PRECEDENCE SCHEDULER [2]
PrecedenceList<jobListInterface>[1-L]
PRECEDENCE SCHEDULER [N]
PrecedenceList<jobListInterface>[1-M]
S1; COBEGIN BEGIN S2; COBEGIN S31; S32; : S3N; COEND S5; END S4; COEND S7;
OPTIMAL SCHEDULER[1][1]
OPTIMAL SCHEDULER[1][2]
OptimalList<taskListInterface>[2]
AgentList<agentListInterface>
OPTIMAL SCHEDULER[1][K]
OptimalList<taskListInterface>[2]
AgentList<agentListInterface>
Agent[1]
Agent[2]
Agent[N]

Fig. 1. A block diagram of the optimization Scheduler

2.1 A Priority Scheduler.

This scheduler uses a modified cyclic Round-Robin queue. The priority of jobs is managed by a time-slice t_q, which is assigned to a particular job.

The t_q determines the activity for a specific task. After the expiration of this period, the current job is paused and control is passed to the next job. A Round-Robin mechanism of working time is not prone to neglect the tasks with lower priority (starvation process).

Each job has a guaranteed periodical running time. The quality of proposed scheduling strategy is depending on the determination of the optimal time period t_q for the task context switching.

A short period of time switching represents a significant increase of load on the system. A long interval of time t_q degrades Round-Robin scheduler to the FCFS queue. The timetable of FCFS scheduler is inefficient in this case, see Sec.(4.1 Priority Scheduler).

2.2 Role of the Precedence Scheduler

The precedence scheduler is responsible for managing the data flow in the individual tasks. Performs the role of Mid-Term scheduler for a current task.

This scheduler is designed as a priority FIFO queue with feedback. The scheduler monitors the progress of the running tasks in the currnet job which provides the results for the next tasks. Solves a generic acyclic graph (acyclic Directed Graph), which describes the workflow of the solved task. Scheme of the precedence graph is defined by the user mandatory.

The precedence scheduler also solves check-pointing of the system. When an exception occurs, the system can restore the status before failure. Subsequently this defective part is sent to the processing again. Is able to solve only a transient type of errors. These errors are defined as temporary and correctable errors (recoverable errors). Can not solve permanent types of errors. These errors mean failure of the entire system, including the schedulers and also the server.

2.3 The Optimalization scheduler

The optimization scheduler is designed for the finding the optimum distribution of computational load for each computing agents connected to the grid.

The aim is to minimize the computational time - makespan. Makespan is defined as the time difference between the start and end of a sequence of jobs in an environment of the independent parallel machines. The makespan is a good indicator for the throughput of the Grid Services.

The optimization scheduler together with a priority scheduler solves the problem of dynamic scheduling. This is applied during adding another task to an already running process. It also applies in a case of failure of one of computing resources. The first input parameter for optimizing scheduler is the capacity of computing resources. The capacity is evaluated by the system for all computational agents, see Sec.(3- Evaluation of the Capacity of Computing Resources).

A second input value is the duration of the task. This value is entered by the user when the job is started. It is a mandatory input parameter.

3 Evaluation of the Capacity of the Computing Resources.

The input variables for the evaluation of the computational capacity:

- *Memory size:* Capacity of the memory .
- *network throughput:* Rate the data transmission speed in the network.
- *Properties of the computing units:* The CPU and GPU parameters.

The variables described above are the basic parameters for the test and determine the performance capacity of the computational resources. For the benchmark test is designed a simple transcendental equation $cos(x) - x = 0$, which can be solved by iteration.

In the first phase, the performance of CPUs is measured. The algorithm starts with the number of parallel threads based on the number of CPU cores. The presence of the GPU is also included in the determination of capacity the computing resources.

The algorithm for evaluation of performance computing resource was chosen according to the following criteria. The number of tasks running in parallel must not exceed the number of processor cores. The CPU computational capacity is an indicator of the performance characteristics of computing resources. The GPU computational capacity gives the possibility of using the graphics card for a special parallel tasks.

For evaluating and comparing the performance of sources requires a certain standard. Based on this standard is evaluated the measured values of any other sources. For this standard was chosen normal office computer with a standard power (CPU) + memory without a graphics processing unit (GPU). The performance was measured by the algorithms described above. This standard was evaluated by the lowest scores from the set $C_{cpu} = (10, \ldots, 24), C \in N$.

The capacity of the GPU was empirically described by the set $C_{gpu} = (2, \ldots, 8), C_{gpu} \in N$. Calculations on the GPU requires a different approach (technology CUDA, OpenCL, OpenGL).

The measurement shows that a significant impact on the capacity of the system are the I/O operations. It is evident in Fig. 3 and the Hill Climbing algorithm sorted section. It was suggested the optimal distribution of computing capacity. Sorting by the length of time duration was assigned to some of the computational agents, a large number of the tasks with short run-time.

This has a negative impact on processing time. In practice, it has proven advantageous the random distribution of tasks with different length processing. Load based on the I/O operations is better distributed.

Evaluation of the Computational Capacity

	GPU is detected				GPU not detected			
CPU time [ms]	1 000	500	250	125	1 000	500	250	125
C_{CPU}	10	12	13	14	10	12	13	14
GPU time [ms]	100	50	25	15				
C_{GPU}	2	4	6	8				
$C = C_{CPU} + C_{GPU}$	**12**	**16**	**22**	**24**	**10**	**12**	**13**	**14**

Table 1. Capacity of the Computing Resources

4 Definition of the Time Interval of the Process Scheduling.

4.1 Priority Scheduler

Priority scheduling is designed as cyclic (Round Robin) queue.

Input parameters:

N	The number of tasks running concurrently.
t_n	The predicted time of the job run - specified by the user.
$P \in (1, \ldots, 5)$	Priority of the task - specified by the user.
t_q	Time interval for the job switching.

Output Parameters:

t_J	The time allocated for the task.

The time interval that is allocated for the running job affects the behavior of the priority scheduler[7]. A selection of short time interval significantly increases the system load due to the context switching and also increases the risk of starvation processes. Starvation may occur so that the following process has no computing resources at time t. They may be busy with other processes. A long interval limit degrades cyclic queue to the FCFS. It will handle tasks sequentially. This condition has a negative impact on the optimization of processing time.

Time interval for the job switching

$$t_q = \frac{\sum_{n=1}^{N} t_n}{N} \tag{1}$$

The time allocated for the task

$$t_j = t_q \left(\frac{1}{P_{max} - P + 1} \right) \tag{2}$$

The Exception - starvation process is solved by increasing the priority tasks on priority $P+1$. A subsequent operation returns the priority tasks to its original value.

5 The Optimization Scheduler

The optimization scheduler together with precedence scheduler solves the distribution of parallel and sequential of tasks on each computer's agent for the current job. The aim is to minimize the makespan processing tasks in an identical computational machine.

Input parameters:

$m = (1, \dots, M)$	The number of the computational agents.
$j = (1, \dots, N)$	Set of the tasks running concurrently.

Output parameters:

$t = t_{scd} + l_i$	Run time of the job = time scheduling + makespan.

The batch job i, consumes t_{ij} units of time. Load of the computing agent is

$$l_i = \sum_{(j \in J_i)} t_{i,j} \tag{3}$$

and

$$l_{max} = \max_{(i \in m)} l_i \tag{4}$$

is the maximum load.

The value l_{max} is called a makespan of the job. In this case, can be said that grid computing agents belong to a set of identical machines. Identical in the sense that the job can be started on any of them. In this case is $t_{i,j} = t_j$ for $i \in M$ and $j \in N$ [3].

The principal task for this scheduler is a minimize of the makespan l_{max}. The criterion for optimizing is job time in the process and the best distribution of load on all computing agents who is available. Total time is the sum of the run time of the tasks in the current job (makespan) and the time needed to create a scheduled task. This fact must be included in the design of optimization criteria.

$$t = t_{scd} + l_i \tag{5}$$

The proposal of the parameters of the scheduler is dependent on two conflicting values. It is necessary to find their optimal size. The quality of the scheduling process is depending on the algorithm and time during which is running.

For the design of appropriate parameters were studied some metaheuristic algorithms. The main task was to find the most appropriate algorithm and determining a reasonable time interval t_{scd} for the scheduling process. The time period for finding the optimal solution has a significant impact on the overall processing time. It is necessary to compromise between quality the proposed plan and time to create a suboptimal schedule. Run time of the schedule was determined from the total task time measurement. It was measured the total time for 1 000, 3 000 and 10 000 tasks. The time interval for scheduling was 0, 10, 60 and 300 seconds.

Five measurements were performed for each type of algorithm and the time interval. Fig. 2 shows the average values of these measurements. Time for scheduling in the length of 0 seconds downgraded optimization schedule to the FCFS.

The optimal time is derived from the measured values. The optimum time is deducted in Fig. 3. The values t_{scd} for the individual algorithms were not much different. Using linear regression has been found the dependency - a straight line which describes the relationship between a number of the scheduled tasks and time t_{scd} for the scheduling with sufficient accuracy.

Run time of scheduling algorithm

The number of tasks in the current job	1 000	3 000	10 000
		$t_{scd}[s]$	
Strategic Oscillation	30	60	140
Step Count Hill Climbing	30	60	150
Tabu Search	25	60	140
Simulated Annealing	30	70	135

Table 2. Run time of scheduling algorithm $t_{scd}[s]$

Relation between the number of task in the current job and time for the scheduling

$$t_{scd} = 0.0122J + 20.69 \tag{6}$$

During the measurement was evaluated run time of elaboration for different optimization algorithms. Measurements were carried out so that each optimization algorithm had the task to propose a timetable for 1 000 tasks. For each algorithm, was performed five measurements. The diameter measurements were evaluated. The best results were achieved with the Late Acceptance algorithm. Poor results were measured with the hill climbing algorithm. The result was that for some agents were assigned to a large number of short tasks. Thee computing nodes have been burdened with a large overhead I/O operations. Algorithms with random task run length showed much better results.

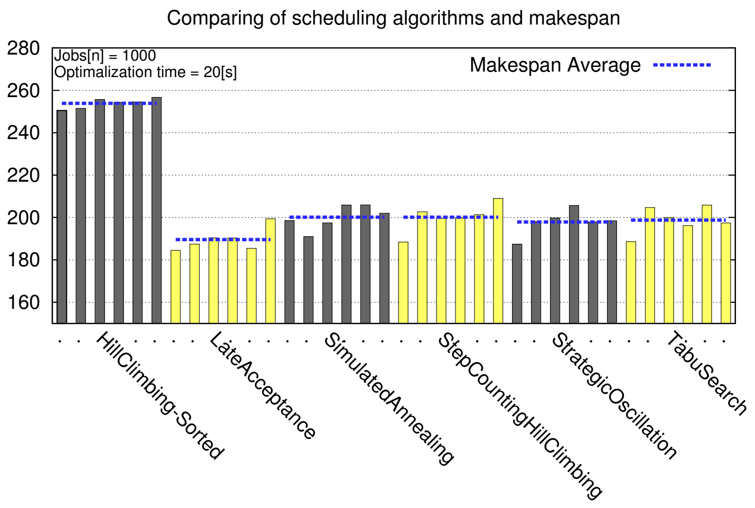

Fig. 2. Comparing of scheduling algorithms and makespan

6 Conclusion

The aim of this work was the description and design of appropriate strategies and parameters in the grid computing environments to optimize performance, response time and throughput of the service. It was shown that for solving job shop Scheduling type which belong to a category of NP - complete problems it is necessary to select some compromises between quality results and a total length of treatment. The dependence on the quality of the final plan and the time required for calculation defines the point that determines the threshold at which further improve the quality of the final timetable of the plan is ineffective. The proposed approximation dependence of scheduled tasks and time of the runtime scheduling algorithm t_{scd} see (6), describes this dependence well. Approximation proposal does not address the limitations resulting from Amdahl's law [4].

Segmentation into three separate objects priority precedence and optimization has led to simplify the design of the scheduler.

Measurements showed that the effect of time for task scheduling has similar results for all investigated optimization algorithms. For all types of algorithms that were examined, it is possible to use the same methodology to determine the length of time (t_{scd}) for the scheduling algorithm.

The aim of future work is to verify the properties of other stochastic optimization algorithms based on the principle of evolutionary strategies. Particularly Ant Colony Optimization, which observes the behavior of ants in search of food. Ant Colony Optimization well simulates the concept of finding the shortest path. The advantage of this algorithm is less sensitive to premature convergence to the insignificant local extremes.

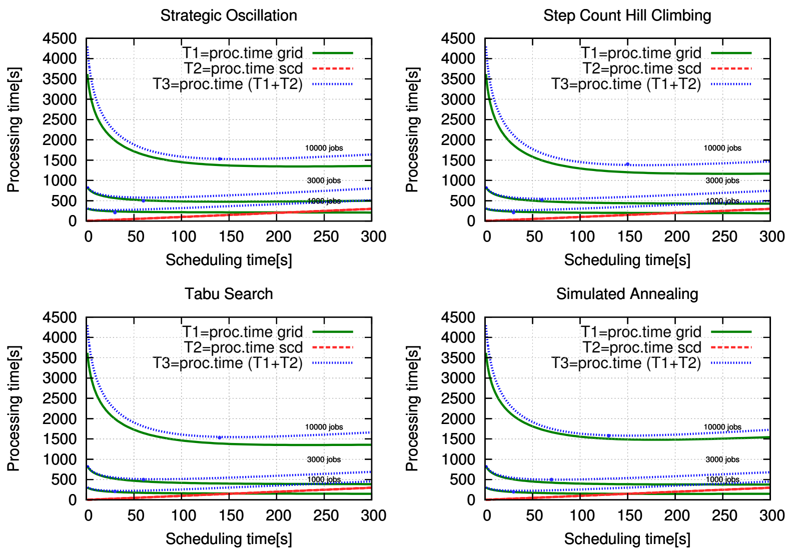

Fig. 3. Measured values of the makespan

References

1. Ali Anjomshoaa, EPCC, Fred Brisard, CA, Michel Drescher, Fujitsu, Donal Fellows, UoM An Ly, CA Stephen McGough, LeSC Darren Pulsipher, Ovoca LLC Andreas Savva, Fujitsu GFD-R.136 Job Submission Description Language (JSDL) Specification http://forge.gridforum.org/projects/jsdl-wg 28 July, 2008 Copyright (C) Open Grid Forum (2003-2005, 2007-2008). All Rights Reserved.
2. Marty Humphrey, UVA Chris Smith, Platform Computing, Marvin Theimer, Microsoft, Glenn Wasson, UVA JSDL HPC Profile Application Extension, Version 1.0 July 14, 2006 Updated: October 2, 2006 Copyright Open Grid Forum (2006-2007). All Rights Reserved.
3. Souza, A. Combinatorial Algorithms Lecture Notes, Winter Term 10/11, Humboldt University Berlin,
4. Sun, X.-H. and Chen, Y. Reevaluating Amdahls law in the multicore era Journal of Parallel and Distributed Computing , 2010, 70, 183 - 188
5. Ezugwu, A. E. and Frincu, M. E. and Junaidu, S. B. A Multiagent-Based Approach to Scheduling of Multi-component Applications in Distributed Systems, Advances in Intelligent Systems and Computing, Artificial Intelligence Perspectives and Applications, Springer International Publishing, 2015, 347, 1-12
6. Lukasik, P. and Sysel, M. A Task Management in the Intranet Grid, Modern Trends and Techniques in Computer Science; Springer International Publishing,2015, Advances in Intelligent Systems and Computing. Springer International Publishing, 2015, ISBN 978-3-319-18472-2 (Print) DOI 10.1007/978-3-319-18473-9, 2015, 349, 77-85
7. Noon, A. and Kalakech, A. and Kadry, S. A New Round Robin Based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average, IJCSI International Journal of Computer Science Issues, 2011, 8, 224-229
8. Rodero, I.; Guim, F.; Corbalan, J.; Labarta, J., How the JSDL can exploit the parallelism?, Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on , vol.1, no., pp.8 pp.,282, 16-19 May 2006 doi: 10.1109/CCGRID.2006.55 `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=\&arnumber=1630829\&isnumber=34197`
9. Marvin Theimer, Microsoft Corporation, Chris Smith, Platform Computing Corporation An Extensible Job Submission Design May 5, 2006 Copyright (C) Global Grid Forum (2006). All rights reserved, Copyright (C) 2006 by Microsoft Corporation and Platform Computing Corporation All rights reserved.